UCRL-PRES-209446

Algorithmic Issues for Scaling Structured AMR Calculations to Thousands of Processors

Andrew Wissink

with

Brian Gunney, David Hysom, Richard Hornung

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

February 14, 2005
SIAM CSE05

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Center for Applied Scientific Computing

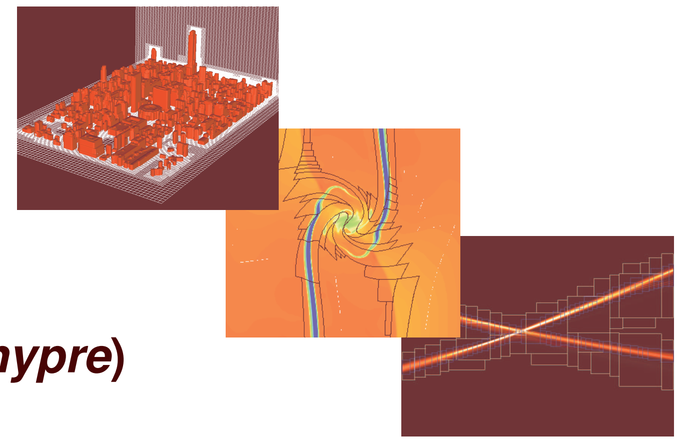
Outline

- Structured Adaptive Mesh Refinement (SAMR) overview
- Parallel implementation approaches used in SAMRAI
- Scaling issues on O(1000) processors
- Predictions of scaling issues on O(100,000) processors

SAMRAI

Structured Adaptive Mesh Refinement Application Infrastructure

- **SAMRAI provides parallel AMR support to applications**
  - **High-level reusable AMR algorithms (e.g. timestepping, dynamic grid generation)**
  - **Parallel support (MPI)**
  - **Parallel tools (VAMPIR, TAU)**
  - **Checkpointing & restart support (HDF)**
  - **Interfaces to solvers (PETSc, PVODE, *hypre*)**



Current SAMRAI users regularly run on large processor systems

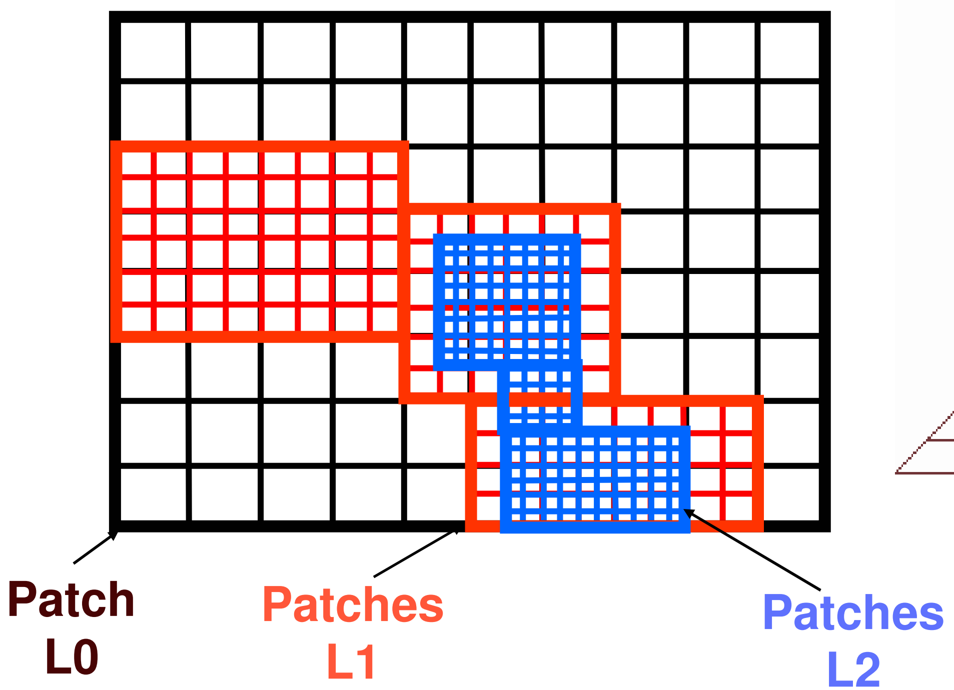
MCR Linux cluster

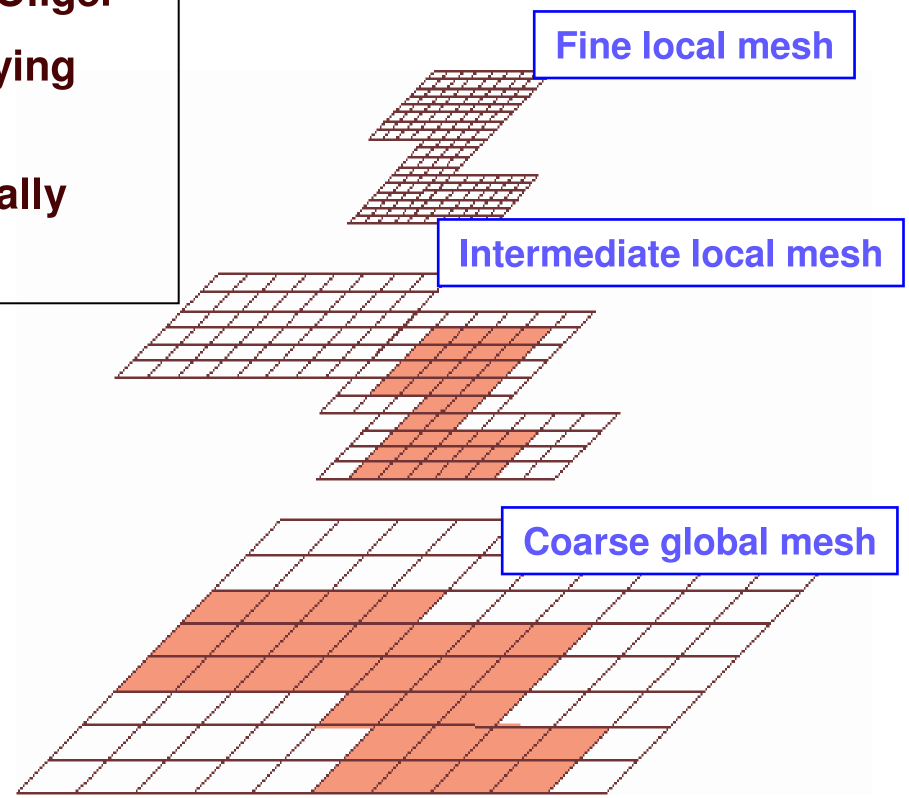
IBM Blue Pacific

TC2K Alpha cluster

Structured AMR (SAMR) employs a dynamically adaptive "patch" hierarchy

- Based on methods of Berger, Colella, Oliger
- Hierarchy defines nested levels of varying mesh resolution
- Data stored on patches covering logically rectangular index space



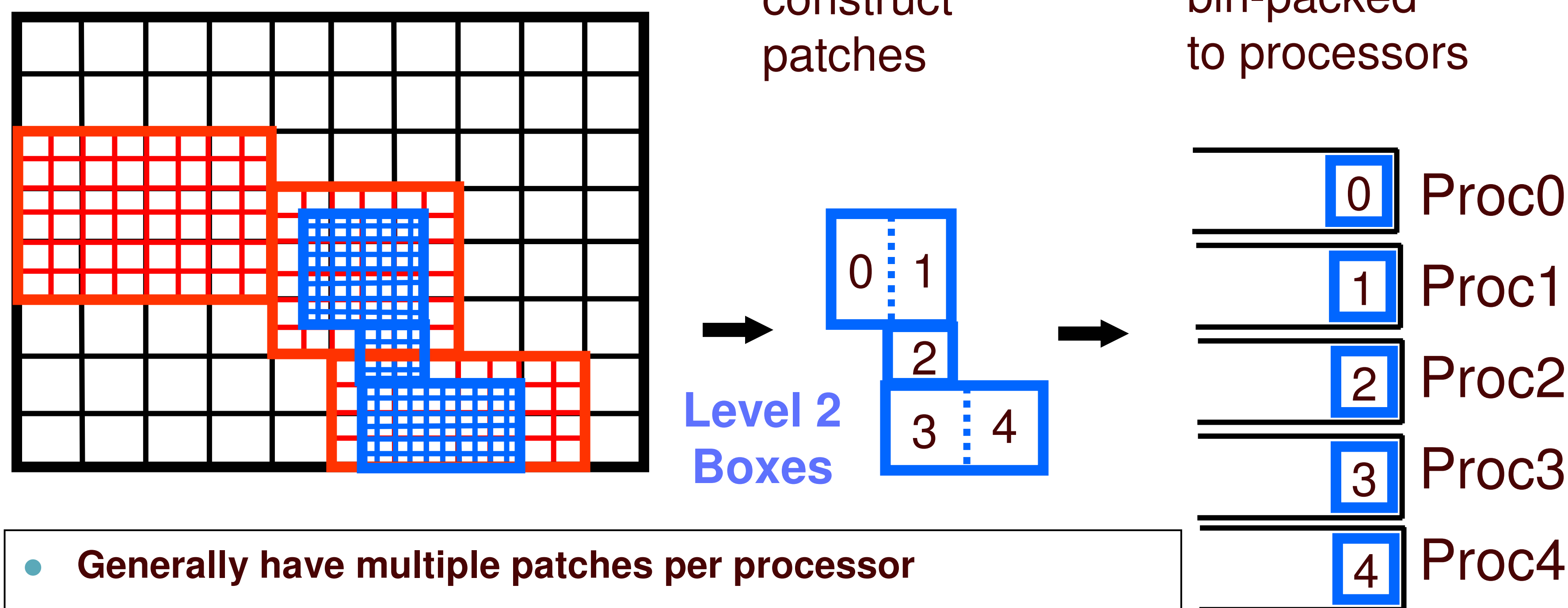


Patches distributed to processors to balance computational workload

1) Box regions constructed

2) Boxes split to construct patches

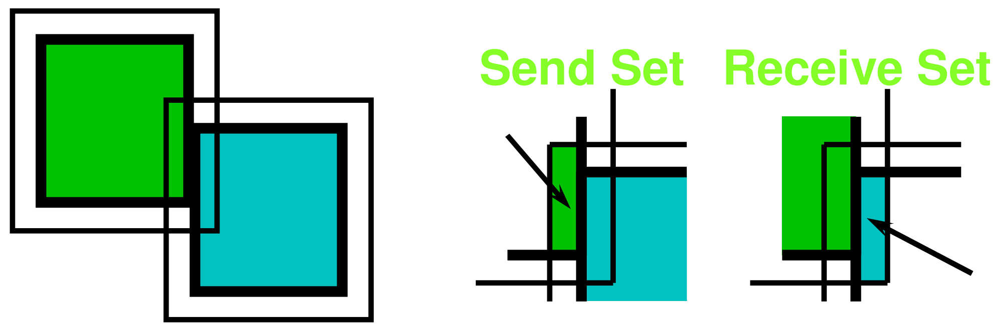
3) Patches bin-packed to processors



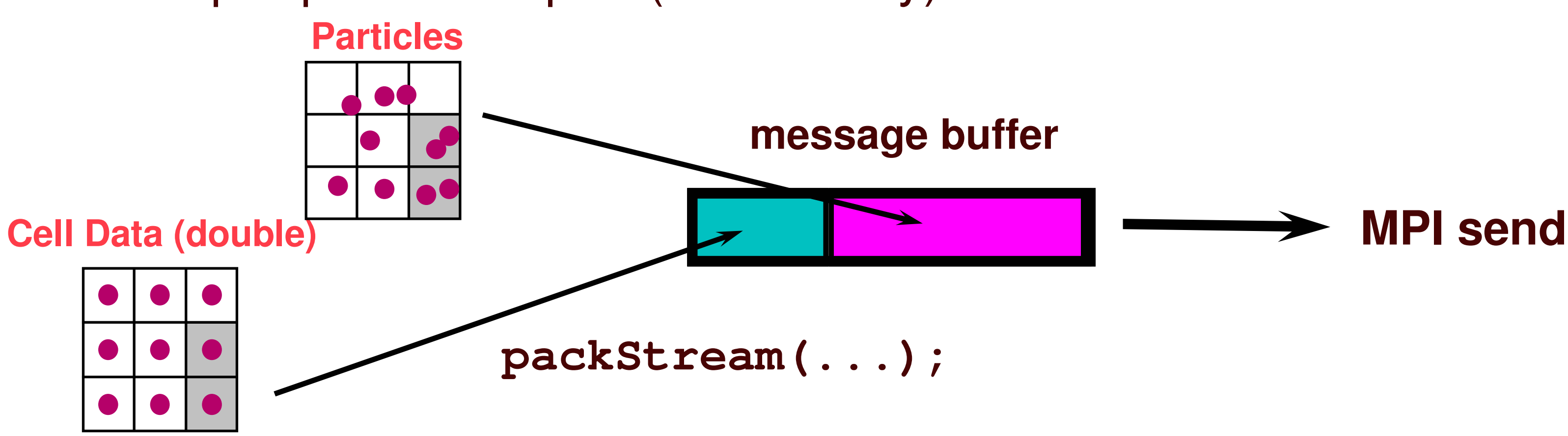
- **Generally have multiple patches per processor**
- **Each level load balanced separately**
- **Spatial bin packing may be used to maintain locality of patches on processors**

Communication schedules create and store data dependencies

- **Amortize cost of creating send/receive sets over multiple communication cycles**

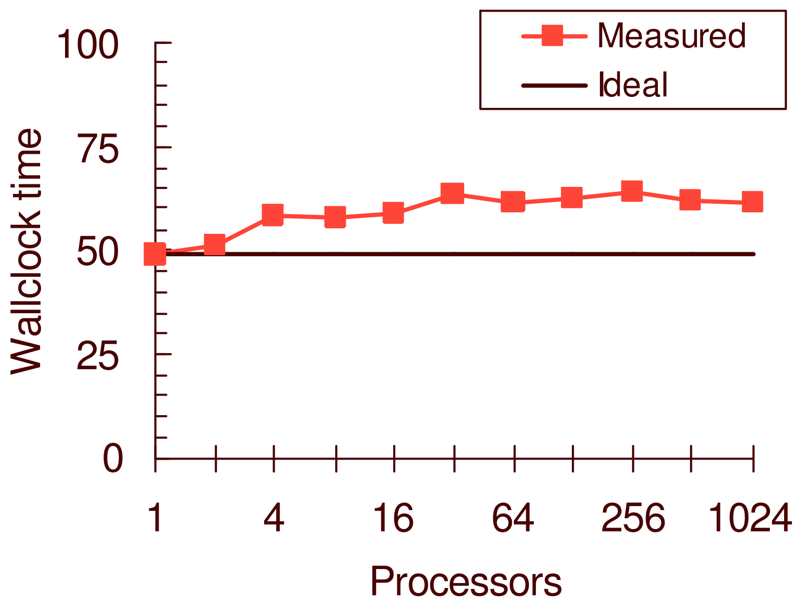


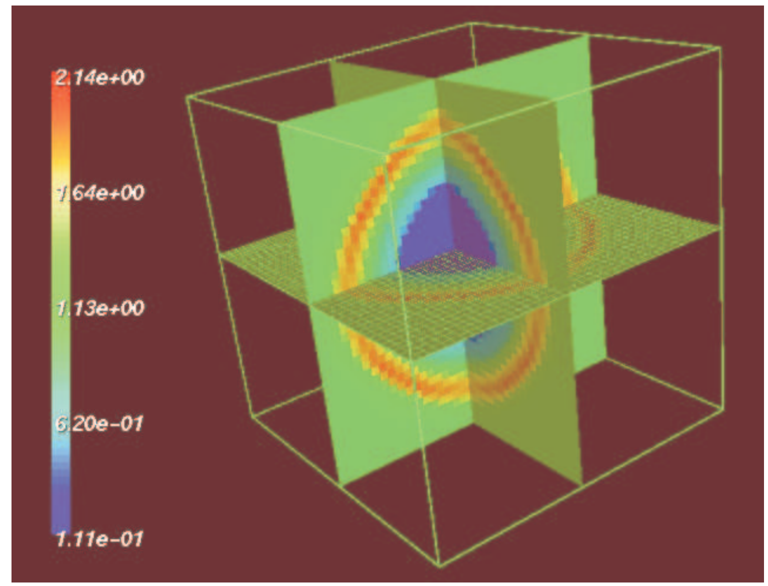
- **Data from various sources packed into single message stream**
  - supports complicated variable-length data
  - one send per processor pair (low latency)



Non-adaptive calculations using SAMRAI show good scaling

Scaled speedup for non-AMR hydro calculation





- Majority of computational spent in patch integration routines. Library code < 5% of total wallclock time

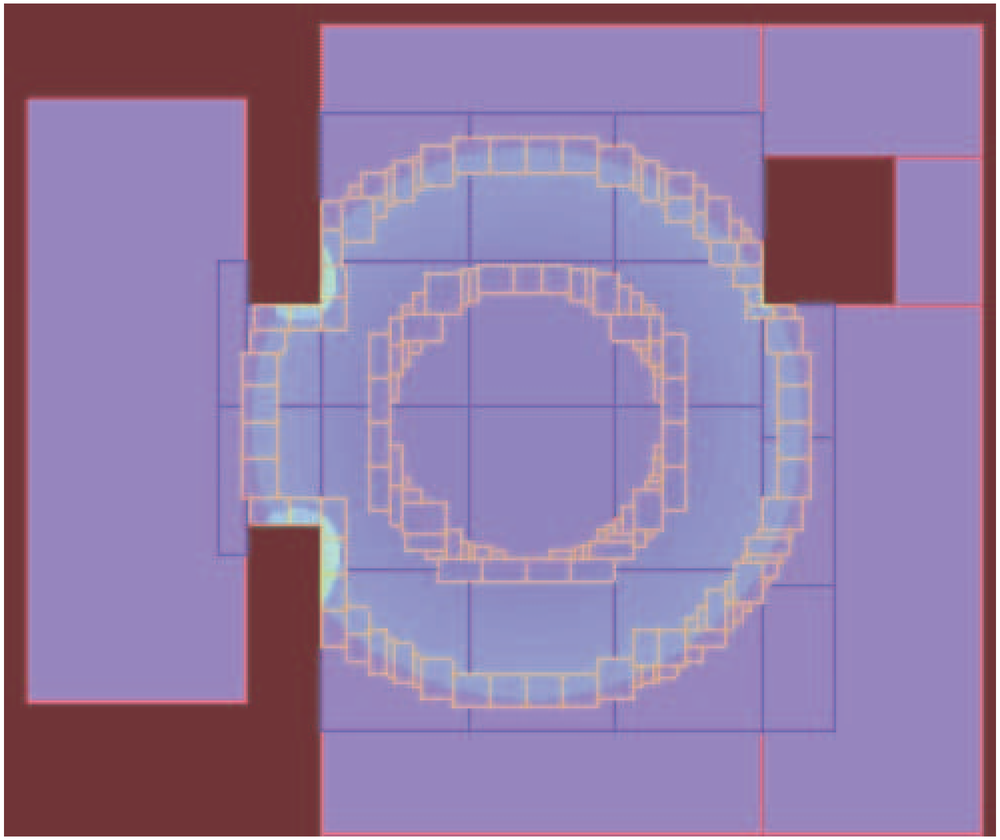
Dynamic mesh adapts to features as solution evolves

Adaptive solution of Euler equations

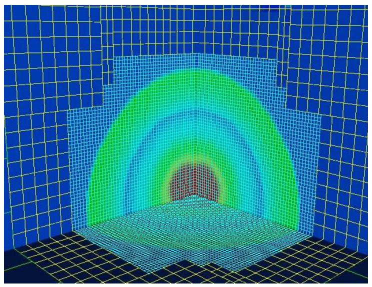
Initial conditions:

inside sphere
density = 8.0
pressure = 40.0

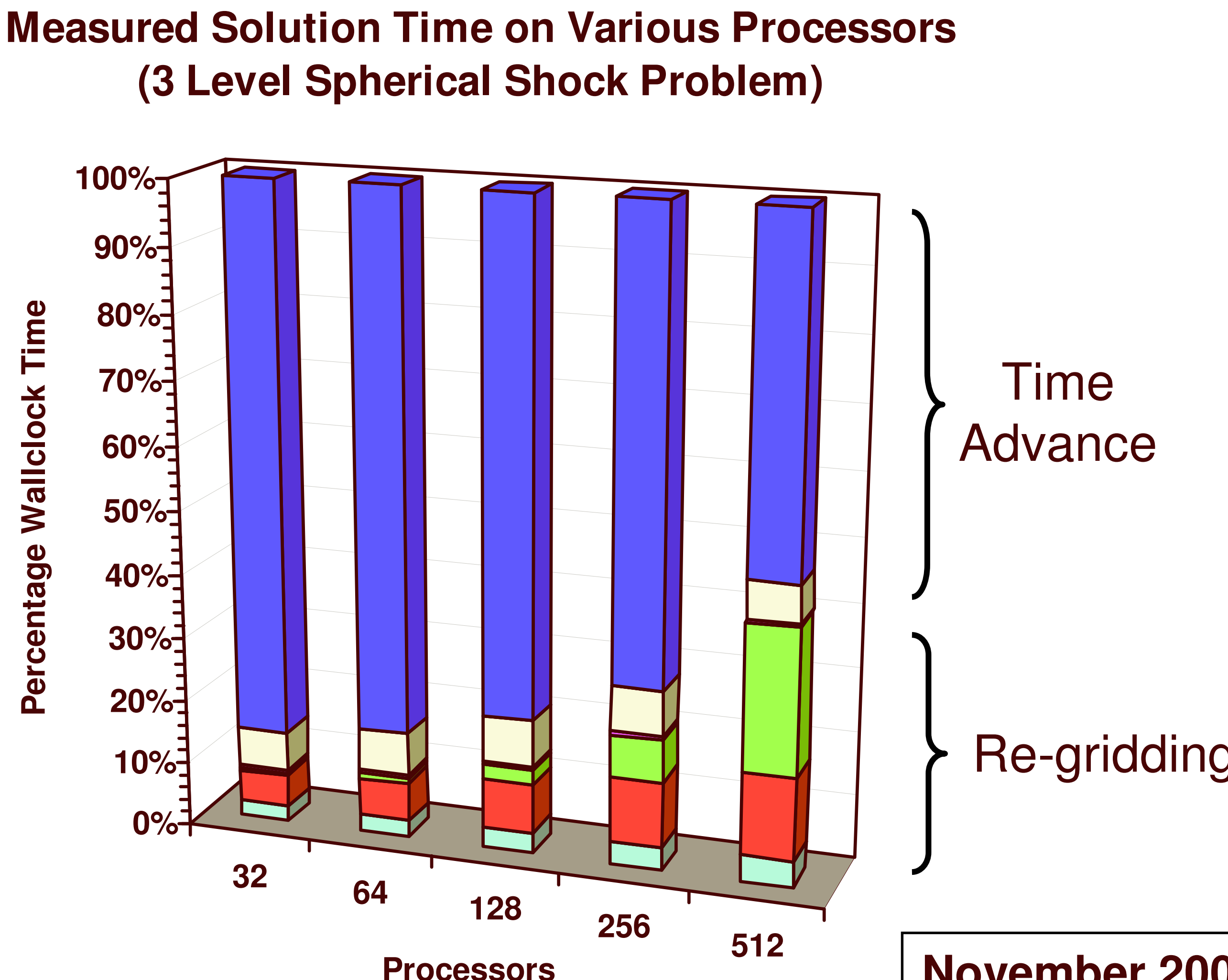
outside sphere
density = 1.0
pressure = 1.0



Adaptive problems show poor scaling in dynamic gridding operations



Non-scaled Euler calculation
IBM Blue Pacific

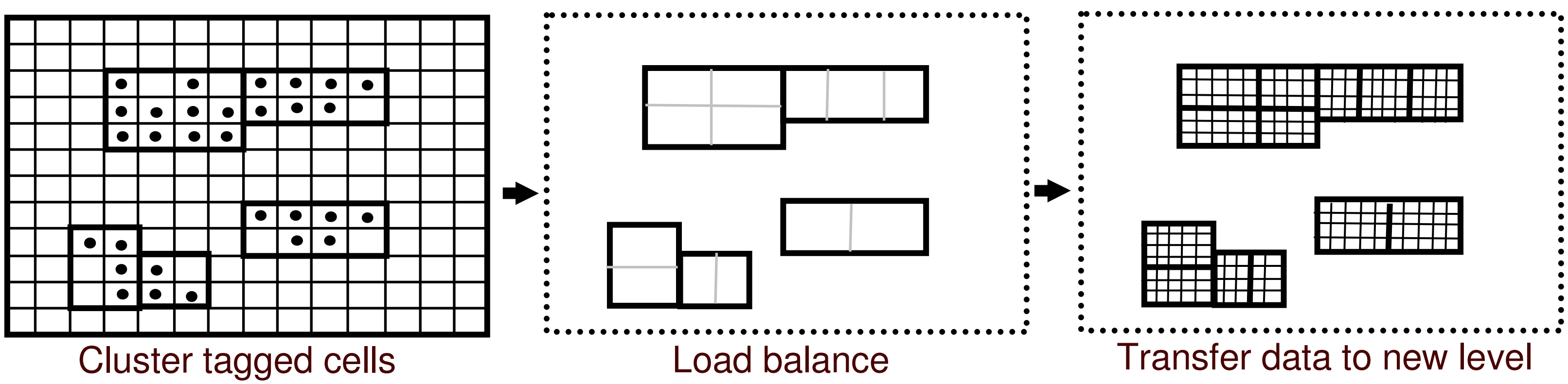


November 2001

Summary of what we mean by “adaptive gridding”

Steps required to construct a new refinement level:

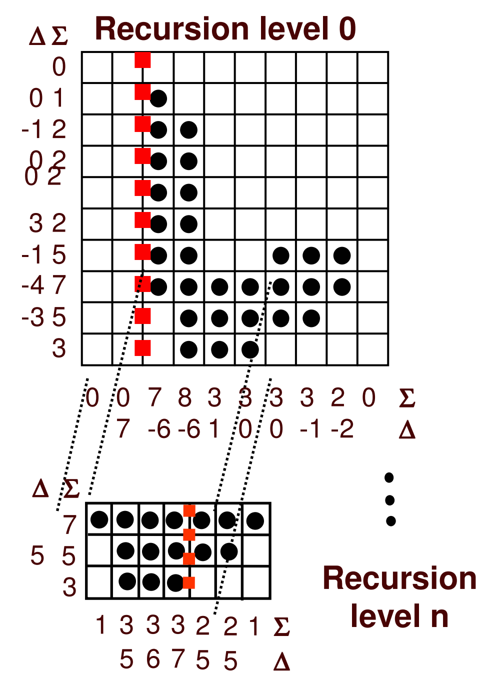
		% Total Time
1.	Tag cells (on coarser level) requiring refinement	< 1%
2.	Cluster tagged cells into “box” regions	1% - 46%
3.	Cut up “box” regions into smaller boxes and determine processor distribution (I.e. load balance)	< 1%
4.	Recompute communication schedules	2% - 87%
5.	Transfer data from old to new level	1-2%



Tagged Cell Clustering Algorithm (Berger Rigoutsos)

- **Original implementation utilized global reductions to construct box histograms**
  - Scales poorly with problem size – number of global reductions grows by $O(n^2)$ (n = number gridcells)
  - Scales poorly with processor count – cost of each global reduction is O(PlogP) (P = number processors)

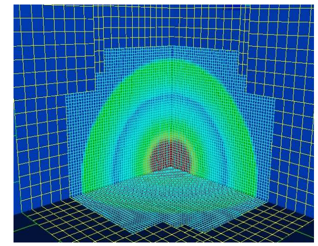
- **Replaced with a new “manager/worker” implementation**
  - Only processors holding tags participated in communication
  - Manager processor accumulates tag histograms and distributes resulting boxes to all processors



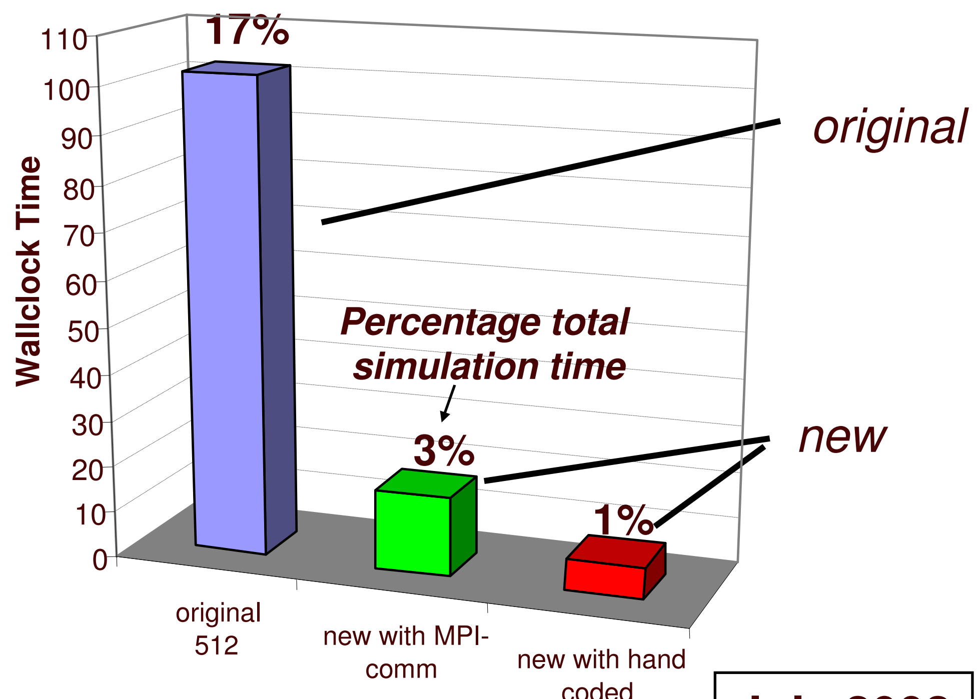
New implementation significantly reduces clustering costs

- **Hand coded MPI send/recvs more effective than MPI communicators**
- **Most significant improvement on systems with slow global reductions**
  - Blue pacific has slower global reductions than newer IBM systems
  - Less significant improvement observed on Linux MCR system

Non-Scaled Euler calculation
IBM Blue Pacific



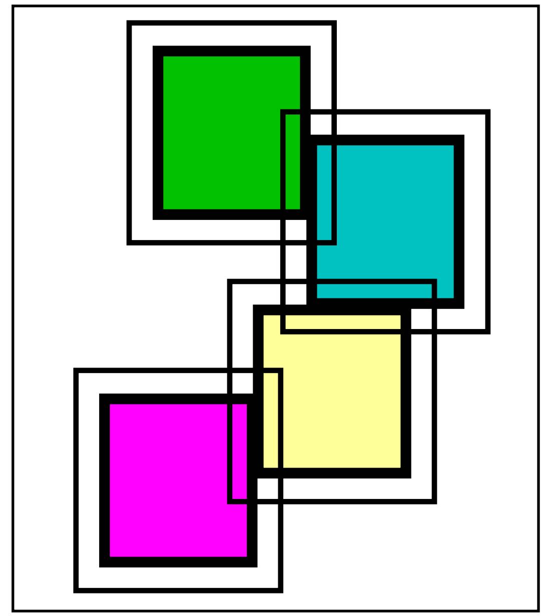
Berger-Rigoutsos – 512 processors



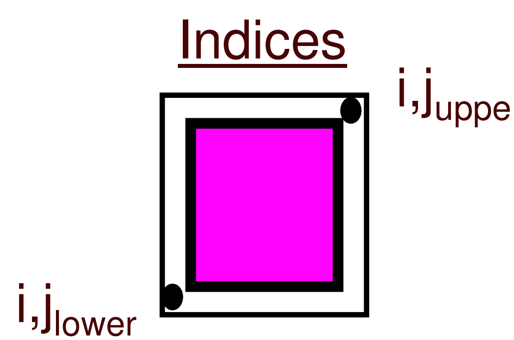
July 2002

Complexity in Comm. Sched. construction becomes significant in large problems

- Data dependencies between patches determined by identifying intersections.
- Original algorithm compared each patch index with all others in the problem.
- Complexity $O(N^2)$ N = number of patches

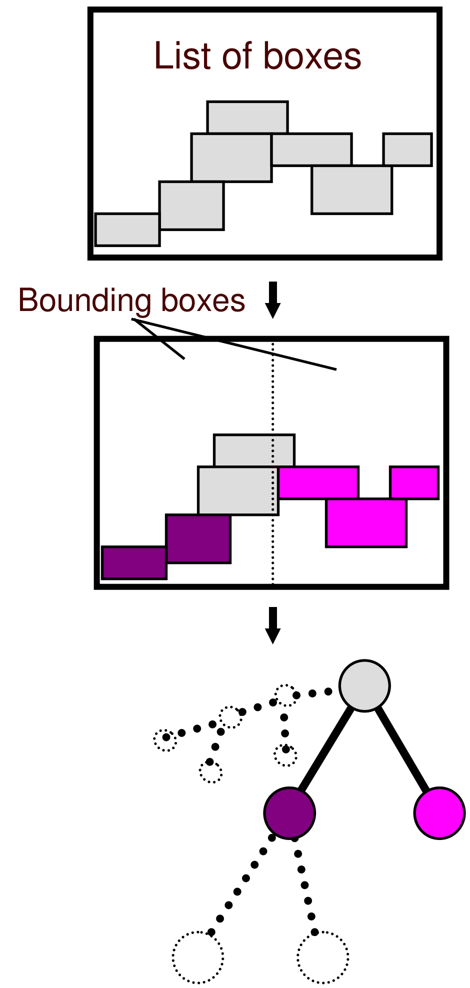


→ ***Communication schedule construction costs grow $O(N^2)$ with problem size***

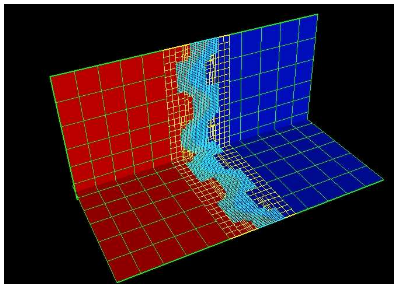


Recursive Binary Box Tree (RBBT) efficiently describes spatial relationships

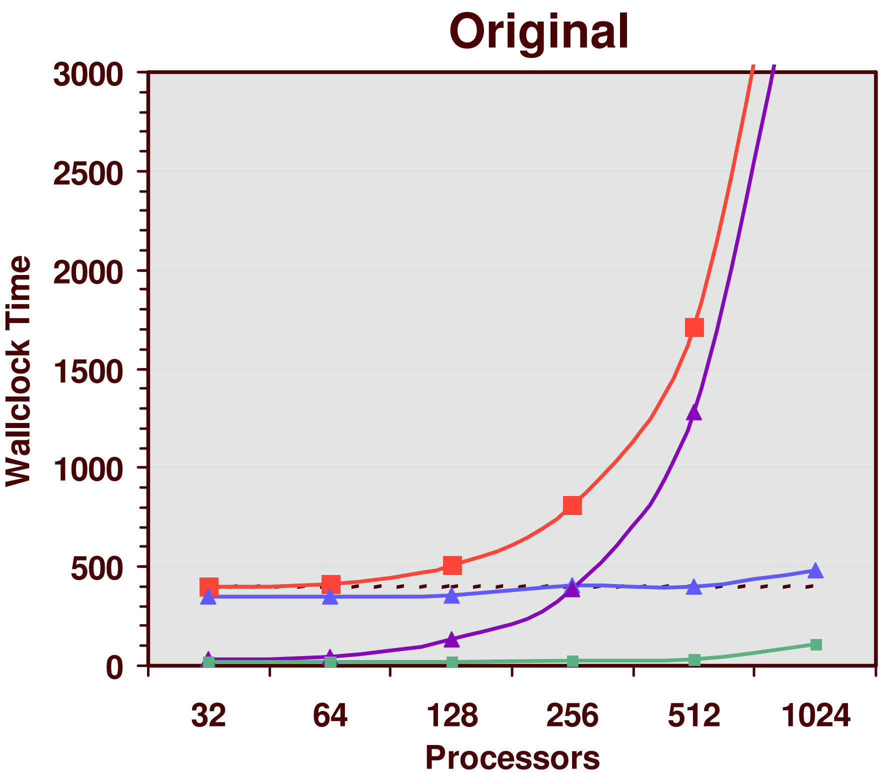
- **Fast determination of box intersection**
- **Analogous to Octree representation**
  - uses bounding boxes and boxlists rather than cells and sub-cells
  - For any given box, determines small subset of boxes that will possibly intersect it, for which we can apply naïve $O(N^2)$ algorithm.
- **Complexity analysis:**
  - Setup: O(N log(N))
  - Query: walk the trees: O(log(N)) per box
  - Runtime complexity: O(N log(N)) – approximate, may vary for different box layouts.

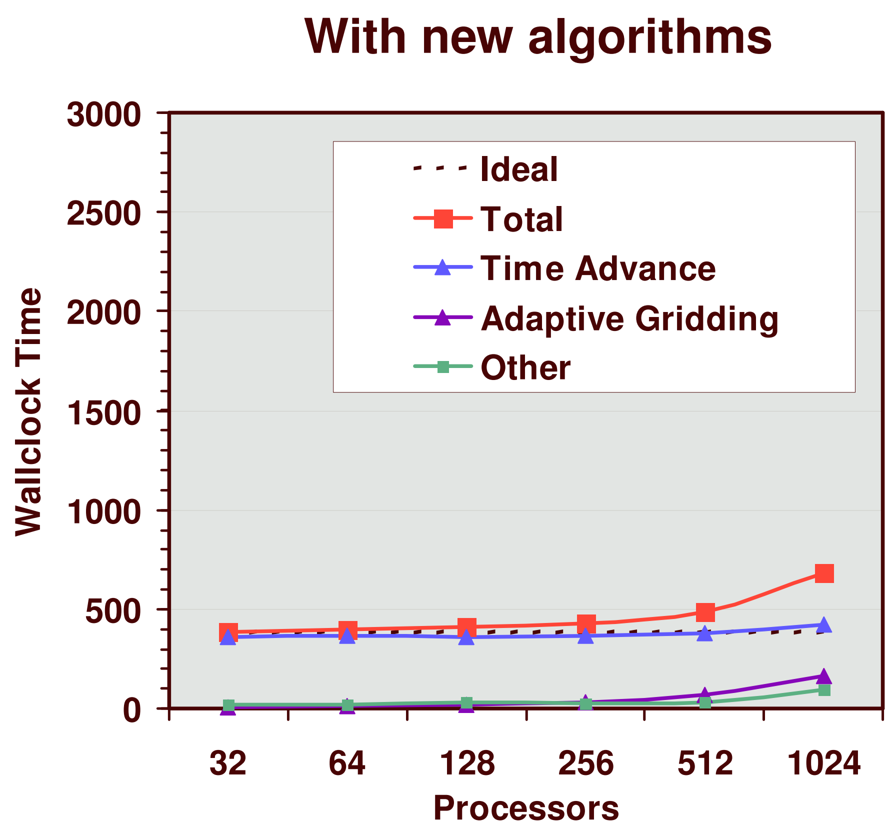


Parallel performance of scaled linear advection benchmark

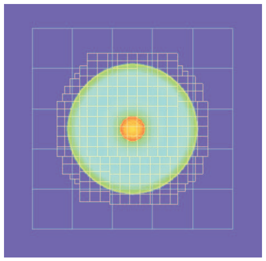


Scaled
3 level linear advection problem
Linux MCR Cluster

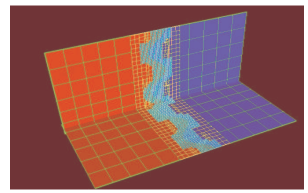




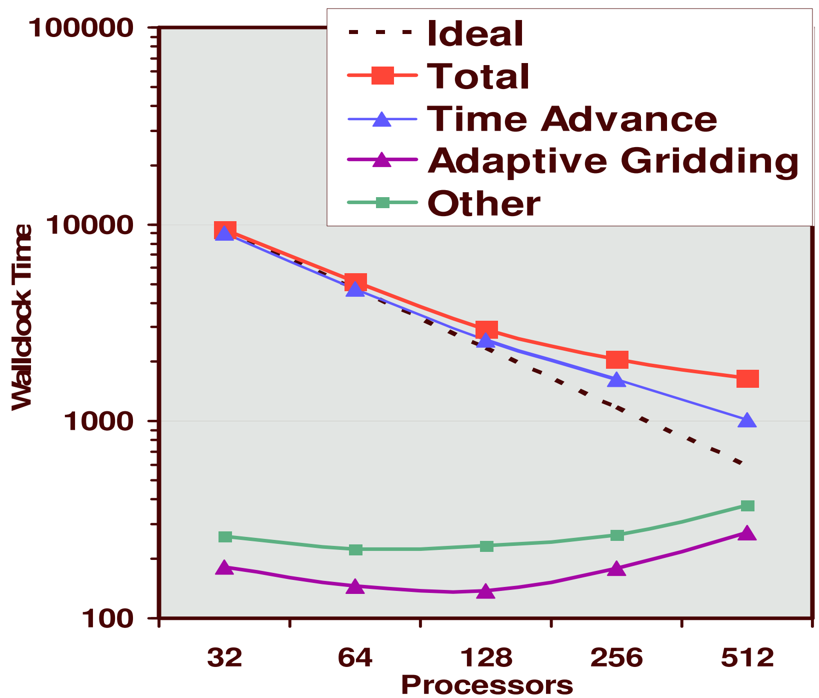
Scaling results after adaptive gridding algorithm modifications

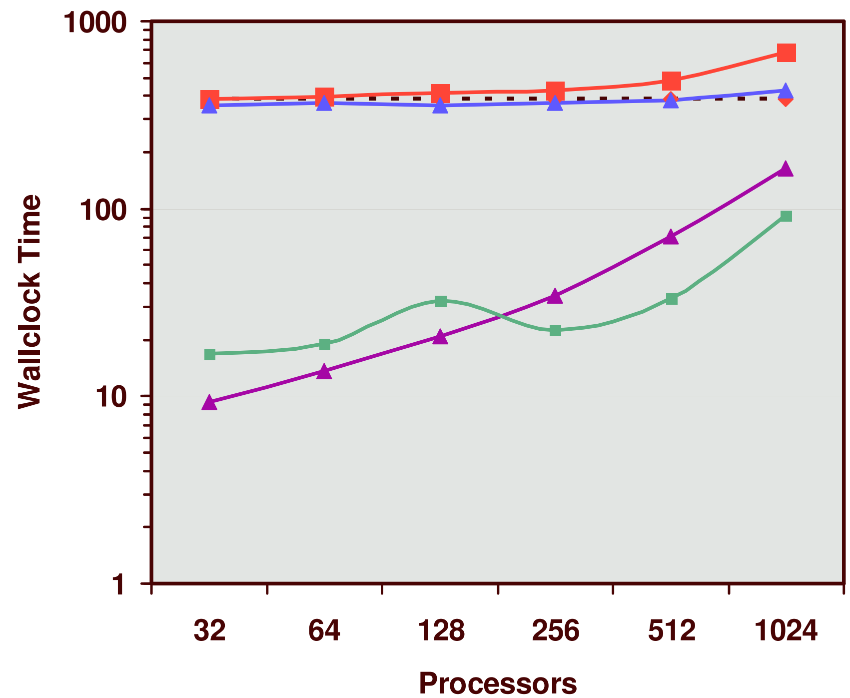


Non-scaled
4 level Euler Problem
IBM Blue Pacific



Scaled
3 level linear advection
Linux MCR Cluster



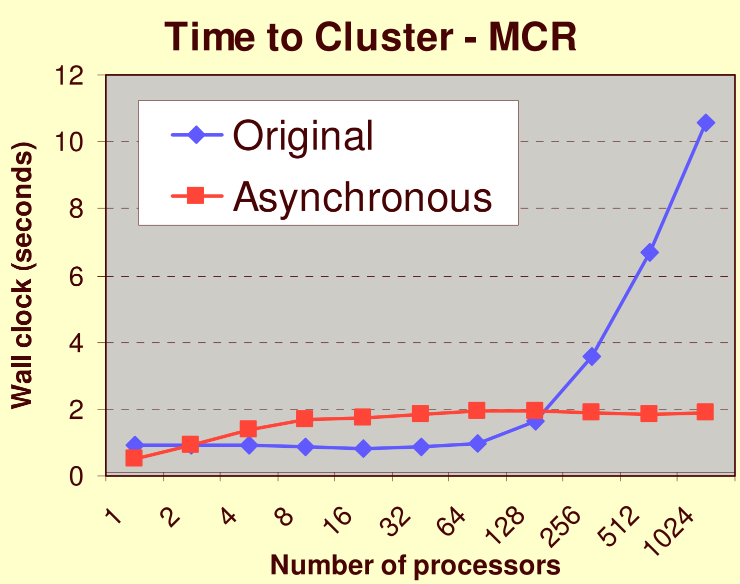


Outline

- Structured Adaptive Mesh Refinement (SAMR) overview
- Parallel implementation approaches used in SAMRAI
- Scaling issues on O(1000) processors
- **Predictions of scaling issues on O(100,000) processors**

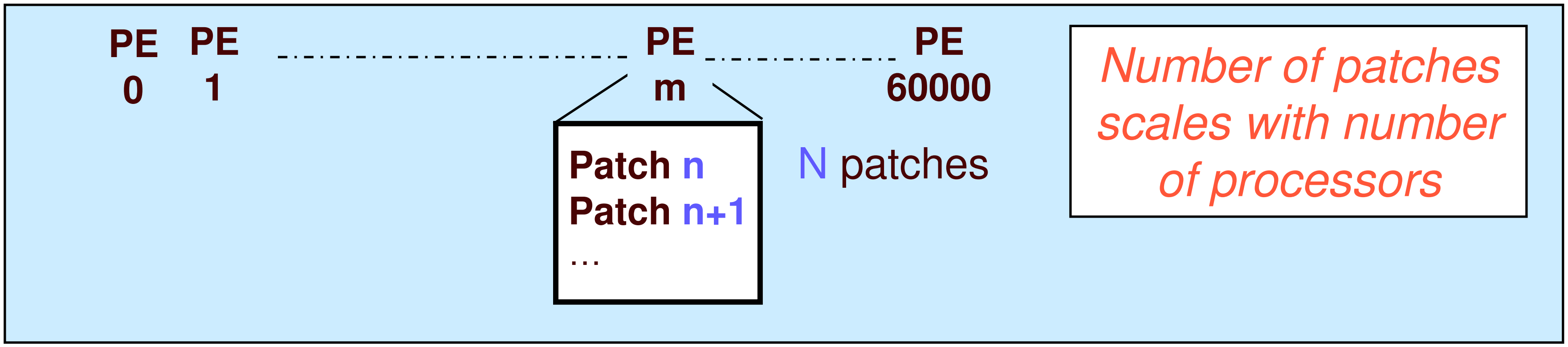
A new asynchronous clustering algorithm for very large scale parallel systems

- **Our new clustering algorithm is effective in reducing costs on O(1K) processors, but not O(10K)-O(100K) processors.**
- **Results from an asynchronous implementation will be presented (B. Gunney – Tues afternoon session CP44)**



Asynchronous implementation avoids global communication and synchronization

More efficient graph-based algorithms required for O(10K)-O(100K) procs



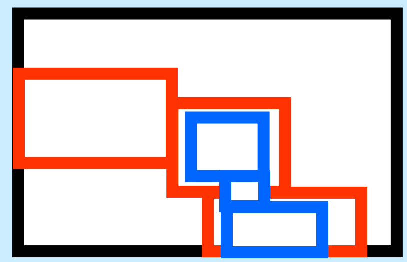
- Naive implementation of box operations in gridding may invoke O(N^2) algorithms (e.g. former communication schedule algorithm).
- We've developed more efficient graph-based algorithms that work on up to O(1000) processors, but ***further work will be required***
- Difficult to assess beforehand because complexity is generally problem dependent.

Storage of globally-known information may introduce memory issues

- **Current approach: Patch lower/upper indices (i.e. “box”) known <u>globally</u> by every MPI process (to determine data locality, communication dependencies, etc.)**

requires consistency across processors: e.g.

```
BoxList boxes = level->getBoxes();
```



- **Because # patches grows with # processors, trivial overhead may become non-trivial on very large scale parallel systems**

procs	patches	per processor storage (MB)
0.5K	2.5K-10K	< 1 MB
60K	300K-1200K	20-80MB

Large overhead for nodes of BG/L

Concluding Remarks

- **Fully adaptive calculations are scalable to O(1000) processors**
- **Adaptive gridding costs are our largest source of parallel inefficiency**
  - Communication is cheap and scales well
  - Re-gridding operations that are trivial on small numbers of processors become significant on large numbers.
  - Tree-based algorithms successful in reducing these costs.
- **Further work required to handle O(100K) processors**
  - New Berger-Rigoutsos clustering algorithm proposed
  - Continued exploration into more efficient tree-based representations of spatial relationships between patches